

ADDLW ADD literal to W [label]: ADDLW k Syntax: 0 ≤ k ≤ 255 Operands: Operation: $(W) + k \rightarrow W$ N,OV, C, DC, Z Status Affected: kkkk kkkk 0000 1111 Encoding: The contents of W are added to the 8-Description: bit literal 'k' and the result is placed in W. 1 Words: 1

Q Cycle Activity:

Cycles:

Q3 **Q4 Q2** Qi Write to W **Process** Decode Read Data literal 'k'

Example:

0x15 ADDLW

Before Instruction 0x10 W After Instruction

0x25 W

ADDWF ADD W to f [label] ADDWF Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ Operation: (W) + (f) \rightarrow dest N,OV, C, DC, Z Status Affected:

Encoding: '

Description:

ffff 0010 01da

Add W to register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register "f (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden

(default).

1 Words: Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ADDWF

0, 0 REG,

Before Instruction

0x17 W 0xC2 REG

After Instruction

0xD9 W REG 0xC2

ADD W and Carry bit to f **ADDWFC** [label] ADDWFC: f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(W) + (f) + (C) \rightarrow dest$ Operation: N,OV, C, DC, Z Status Affected: ffff ffff 00da 0010 **Encoding:** Add W, the Carry Flag and data memory Description: location 'T. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0 Virtual bank will be selected.. If 'a' is 1 the BSR will not be overridden.

Words:

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register T	Process Data	Write to destination

Example:

ADDWFC

REG, 0, 1

Before Instruction

Carry bit = 0x02 REG 0x4D W

After Instruction

Carry bit = 0x02 REG 0x50 W

ANDLW AND literal with W [label] ANDLW Syntax: $0 \le k \le 255$ Operands: (W) AND. $k \rightarrow W$ Operation: N,Z Status Affected: kkkk , 0000-1011 kkkk Encoding: The contents of W are AND'ed with the Description: 8-bit literal 'k'. The result is placed in W. 1 Words: 1 Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to W
<u> </u>			

Example:

ANDLW

0x5F

Before Instruction

0xA3 W

After Instruction

W 0x03

AND W with f **ANDWF** [label] ANDWF f,d,a. Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ Operation: (W) .AND. (f) \rightarrow dest N,Z Status Affected: ffff ffff 0001 01da Encoding: The contents of W are AND'ed with reg-Description: ister 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default). Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

ANDWF

REG, 0, 0

Before Instruction

W = 0x17 REG = 0xC2

After Instruction

W = 0x02 REG = 0xC2

ВС	Branch if Carry

Syntax:

[label] BC n

Operands:

-128 ≤ n ≤ 127

Operation:

if carry bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected:

Encoding:

None 1110 0010 nnnn nnnn

Description:

If the Carry bit is '1', then the program

will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

.

Cycles:

1(2)

Q Cycle Activity:

If Jump:

	Q1	Q2	Q3	Q4
.	Decode	Read literal	Process Data	Write to PC
	No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

BC 5

Before Instruction

PC

= address (HERE)

After Instruction

If Carry

C = address (HERE+12)

If Carry = (PC = 3

PC = address (HERE+2)

BCF	Bit Clear f			
Syntax:	[label] BC	OF f,b	,a	
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$	•		
Operation:	$0 \rightarrow f < b >$			
Status Affected:	None			
Encoding:	1001	bbba	ffff	ffff
Description:	Bit 'b' in reg Virtual band the BSR va will be sele (default).	c will be s due. If 'a	selected, ' = 1, then	overriding the bank
Words:	1			
Cycles:	1	•		
Q Cycle Activity:				
Q1	Q2	Q3_	Q	4
Decode	Read	Proc	ess	Write

Example:

BCF

register "f

7, 0 FLAG_REG,

Data

register "f

Before Instruction FLAG_REG = 0xC7 After Instruction FLAG_REG = 0x47

BN	Branch if	Negativ	e e	
Syntax:	[label] Bi	N n		
Operands:	-128 ≤ n ≤			
Operation:	if negative (PC) +	bit is '1 2 + 2n -		
Status Affected:	None			
-	1110	0110		nnnn

Encoding:

If the Negative bit is '1', then the pro-Description: gram will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	No operation

Example:

HERE

Jump

Before Instruction

PC

address (HERE)

After Instruction

If Negative PC

address (Jump)

If Negative PC

0; address (HERE+2)

Branch if Not Carry BNC [label] BNC Syntax: $-128 \le n \le 127$ Operands: if carry bit is '0' Operation: $(PC) + 2 + 2n \rightarrow PC$ Status Affected: None nnnn 1110 0011 Encoding: If the Carry bit is '0', then the program Description: will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction. Words: 1(2) Cycles: Q Cycle Activity: If Jump: 04

Read literal	Process Data	Write to PC
No operation	No operation	No operation
	'n' No	'n' Data No No

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

BNC Jump

Before Instruction

PC

address (HERE)

After Instruction

If Carry

address (Jump)

address (HERE+2)

BNN

Branch if Not Negative

Syntax:

[label] BNN n.

Operands:

 $-128 \le n \le 127$

Operation:

if negative bit is '0'

 $(PC) + 2 + 2n \rightarrow PC$

0111

Status Affected:

None

Encoding:

1110

Description:

If the Negative bit is '0', then the pro-

nnnn

nnnn

gram will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
30000	'n'	Data	operation

Example:

HERE

BNN Jump

Before Instruction

PC

address (HERE)

After Instruction

If Negative PC

address (Jump)

If Negative

address (HERE+2)

Branch if Not Overflow INV [label] BNV n lyntax: -128 ≤ n ≤ 127)perands: if overflow bit is '0' Operation: $(PC) + 2 + 2n \rightarrow PC$ None Status Affected: 0101 nnnn nnnn 1110 Encoding: If the Overflow bit is '0', then the pro-Description: gram will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction. 1. Words: 1(2) Cycles: Q Cycle Activity: If Jump: Q4 Q3 Q2 Q1 Write to PC Read literal **Process** Decode 'n Data No No No No operation operation operation operation

If N	o Jump:		00	04
	Q1	Q2	Q3	Q4
	Decode	Read literal	Process Data	No operation

HERE Example: Before Instruction

BNV Jump

address (HERE) PC

After Instruction

If Overflow

address (Jump) PC

If Overflow

address (HERE+2) PC

Branch if Not Zero BNZ

Syntax:

[label] BNZ n

Operands:

 $-128 \le n \le 127$

Operation:

if zero bit is '0'

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected:

None

Encoding:

nnnn 1110 0001 nnnn If the Zero bit is '0', then the program

Description: will branch.

> The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

•	Q1	Q2	Q3	Q4:
[Decode	Read literal	Process	No
	5000	'n'	Data	operation

Example:

HERE

BNZ Jump

Before Instruction

PC

address (HERE)

After Instruction

If Zero

address (Jump)

If Zero

=

address (HERE+2) PC

Unconditional Branch BRA [label] BRA n Syntax: Operands: -1024 ≤ n ≤ 1023 $(PC) + 2 + 2n \rightarrow PC$ Operation: None Status Affected: nnnn nnnn 1101 0nnn Encoding: Add the 2's complement number '2n' to Description: the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words:

1

Cycles:

2.

Q Cycle Activity:

Q1	Q2 ¹	Q3	Q4'
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example:

HERE

Jump BRA

Before Instruction

PC

address (HERE)

After Instruction

PC

address (Jump)

Bit Set f **BSF** [label] BSF f,b,a Syntax: 0 ≤ f ≤ 255 Operands: 0≤b≤7 $a \in [0,1]$ 1 → f Operation: None Status Affected: ffff ffff 1000 bbba Encoding: Bit 'b' in register 'f' is set. If 'a' is 0 Vir-Description: tual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value. 1 Words: 1 Cycles: Q Cycle Activity: Q4 Q3 Q2 Q1 Write **Process** Read Decode register "f Data register "f

Example:

FLAG_REG, 7, 1

Before Instruction

FLAG_REG=

0x0A

After Instruction

A8x0 FLAG_REG=

BTFSC	Bit Test F	ile. Skip i	f Clear	
Syntax:	[label] B			
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
Operation:	skip if (f<	b>) = 0		
Status Affected:	None			
Encoding:	1011	bbba	ffff	ffff
Description:	instruction If bit 'b' is fetched du cution is d cuted inst instruction selected, = 1, then	register 'f' is is skipped. O then the ruring the cuilscarded, a ead, makin. If 'a' is 0 overriding the bank wivalue (defail	next instruction of the second	ction uction exe- is exe- o-cycle nk will be alue. If 'a'
Words:	1			
Cycles:	1(2) Note: 3 by	cycles if sl a 2-word	kip and fo	illowed n

Q Cycle	Activity:	

Q1	Q2	Q3	Q4
Decode	Read	Process Data	No
	register 'f'	<u> </u>	operation

If skip:

No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE BTFSC FLAG, 1, 0

FALSE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0

PC = address (TRUE)

If FLAG<1> = 1;

PC = address (FALSE)

BTFSS

Bit Test File, Skip if Set

Syntax:

[label] BTFSS f,b,a

Operands:

0 ≤ f ≤ 255

 $0 \le b < 7$

 $a \in [0,1]$

Operation:

skip if (f < b >) = 1

Status Affected:

i: None

Encoding:
Description:

1010 bbba ffff ffff

If bit 'b' in register 'f' is 1 then the next

instruction is skipped.

If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per

the BSR value (default).

Words:

1

Cycles:

1(2) Note: 3 cycles if skip and followed

by a 2-word instruction

Q Cycle Activity:

Q1	· Q2	Q3	Q4
Decode	Read	Process Data	No
-	register 'P		operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	. No operation
No operation	No operation	No operation	No operation

Example:

HERE

BTFSS

FLAG, 1, 0

FALSE

TRUE

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;

PC = address (FALSE)

fFLAG<1> = 1;

PC = address (TRUE)

Bit Toggle f BTG [label] BTG f,b,a Syntax: $0 \le f \le 255$ Operands: 0≤b<7 $a \in [0,1]$ Operation: Status Affected: None ffff ffff bbba 0111 Encoding: Bit 'b' in data memory location 'f' is Description: inverted. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Q1	Q2	Q3 ⁻	Q4
Decode	Read register "f"	Process Data	Write register 'f'

Example:

BTG PORTC, 4,

Before Instruction:

PORTC = 0111 0101 [0x75]

After Instruction:

PORTC = 0110 0101 [0x65]

BV	Branch if Overflow

Syntax: [label] BV n

Operands: $-128 \le n \le 127$

Operation: if overflow bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding: 1110 0100 mmn nnnn

Description: If the Overflow bit is '1', then the pro-

gram will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

BV Jump

Before Instruction

PC = address (HERE)

After Instruction

If Overflow = 1

PC = address (Jump)

If Overflow = 0

PC = address (HERE+2)

Branch if Zero BZ [label] BZ n Syntax: -128 ≤ n ≤ 127 Operands: if Zero bit is '1' Operation: $(PC) + 2 + 2n \rightarrow PC$ None Status Affected: nnnn 0000 nnnn 1110 **Encoding:** If the Zero bit is '1', then the program Description: will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a

two-cycle instruction.

Words:

1

Cycles:

1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example:

HERE

Jump

Before Instruction

PC

address (HERE)

After Instruction

If Zero

address (Jump) PC

If Zero

address (HERE+2) PC

CALL

Subroutine Call

[label] CALL k,s Syntax:

0 ≤ k ≤ 1048575 Operands:

s ∈ [0,1]

 $(PC) + 4 \rightarrow TOS$, Operation:

 $k \rightarrow PC < 20:1>$

if s = 1

 $(W) \rightarrow WS$,

(STATUS) → STATUSS,

(BSR) → BSRS

None Status Affected:

Encoding: 1st word (k<7:0>)

2nd word(k<19:8>)

kkkk₀ k7kkk 1110 110s kkkk kkkka 1111 k₁₉kkk

Description:

Subroutine call of entire 2M byte memory range. First, return address (PC+4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-

cycle instruction.

Words:

2

Cycles:

2

Q Cycle Activity:

•	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, Write to PC
	No operation	No operation	No operation	No operation

Example:

HERE

THERE, Fast CALL

Before Instruction

PC = Address (HERE)

After Instruction

PC = TOS = Address (THERE)

Address (HERE + 4)

WS = BSRS= BSR

STATUSS = STATUS

Clear f CLRF [label] CLRF f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ $000h \rightarrow f$ Operation: $1 \rightarrow Z$ Status Affected: Z ffff ffff 0110 101a **Encoding:** Clears the contents of the specified reg-Description: ister. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity: Q3 Q2 Q1 Write **Process** Decode Read register 'f' Data

Clear Watchdog Timer CLRWDT [label] CLRWDT Syntax: None Operands: $000h \rightarrow WDT$, Operation: 000h → WDT postscaler, 1 → TO, 1 → PD TO, PD Status Affected: 0000 0000 0100 0000 Encoding: CLRWDT instruction resets the Watch-Description: dog Timer. It also resets the postscaler of the WDT. Status bits TO and PD are set 1 Words: Cycles: Q Cycle Activity: Q4 Q3 **Q2** Q1 **Process** No No Decode operation Data operation

FLAG_REG, 1 CLRF Example: Before Instruction 0x5A FLAG_REG After Instruction

0x00 FLAG_REG

register "f

Before Instruction ? WDT counter After Instruction 0x00 WDT counter WDT Postscaler TO PD 0

Example:

CLRWDT

COMF	Complem	ent f		
Syntax:	[label] C	OMF	f,d,a	
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	5		
Operation:	$(\overline{f}) \rightarrow de$	est		
Status Affected:	N,Z			
Encoding:	0001	11da	ffff	ffff
Description:	The content mented. If W. If 'd' is register 'f' bank will be SSR value be selected (default).	'd' is 0 th 1 the resi (default). De selecte B. If 'a' =	e result is ult is store If 'a' is 0 id, overridi 1, then the	stored in d back in Virtual ng the a bank will
Words:	1			
Cycles:	1			

Q Cycle Activity	•		
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

REG, 0, 0

kampie.	C01.12
Before	Instruction

REG = 0x13

After Instruction

 $\begin{array}{rcl} REG & = & 0x13 \\ W & = & 0xEC \end{array}$

CPFSEQ	Compare f with W, skip if f = W			
Syntax:	[label] C	PFSEQ	f,a.	
Operands:	$0 \le f \le 25$ $a \in [0,1]$	5		
Operation:	(f) - (W), skip if (f) = (unsigned	•	on)	
Status Affected:	None			
Encoding:	0110	001a	ffff	ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).			
Words:	1			'

Cycles:

1(2) Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register "f"	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
operation	operation	Operation	1

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE CPFSEQ REG, 0

NEQUAL

EQUAL

Before Instruction

PC Address = HERE W = ? REG = ?

After Instruction

If REG = W;

PC = Address (EQUAL)

If REG # W;

PC = Address (NEQUAL)

Compare f with W, skip if f > W **CPFSGT** [label] CPFSGT f,a Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ (f) - (W),Operation: skip if (f) > (W)(unsigned comparison) None Status Affected: ffff 0110 010a ffff **Encoding:** Compares the contents of data memory Description: location 'I' to the contents of the W by performing an unsigned subtraction. If the contents of 'f are greater than the contents of W then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1(2) Cycles: Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

-	Q1	Q2	Q3	Q4
-	Decode	Read	Process	No
	20000	register "	Data	operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1 ·	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

CPFSGT REG, 0

NGREATER

GREATER

Before Instruction

PC

Address (HERE)

W

?

≤

After Instruction

If REG

W:

W:

PC

Address (GREATER)

If REG

PC

Address (NGREATER)

CPFSLT	Compare f with W, skip if $f < W$		
Syntax:	[label] CPFSLT f,a		
Operands:	0 ≤ f ≤ 255		

 $a \in [0,1]$

Operation:

(f) - (W),skip if (f) < (W)

(unsigned comparison)

Status Affected:

None

Encodina:

0110	000a	ffff	ffff

Description:

Compares the contents of data memory location 'T to the contents of W by performing an unsigned subtraction. If the contents of 'T are less than the contents of W, then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected. If 'a' is 1 the BSR will not be overridden (default).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register 'f'	Data	operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE

CPFSLT REG, 1

NLESS

LESS

Before Instruction

Address (HERE) PC W

After Instruction

If REG

W: Address (LESS)

PC

W: If REG ≥

Address (NLESS) PC

DAW	Decimal A	djust W	Registe	
Syntax:	[label] DA	/W:		
Operands:	None			
Operation:	If [W<3:0>>9] or [DC = 1] then (W<3:0>) + 6 → W<3:0>; else (W<3:0>) → W<3:0>; If [W<7:4>>9] or [C = 1] then (W<7:4>) + 6 → W<7:4>; else (W<7:4>) → W<7:4>;			
	(W<7:4	>) → w<	:7:4>;	
Status Affected:	<u>c</u>			
Encoding:	0000	0000	0000	0111
Description:	DAW adju- resulting fi variables (and produ- result.	rom the each in p	artier addit acked BC	ion of two D format)
Words:	1			
Cycles:	1			

G C	CIE ACTIVITY	Q2	Q3	Q4	
	Decode	Read register W	Process Data	Write W	
Exa	mple1:	DAW			

Before Instruction

W = 0xA5 C = 0 DC = 0

After Instruction

W = 0x05 C = 1 DC = 0

Example 2:

Before Instruction

W = 0xCE C = 0 DC = 0

After Instruction

W = 0x34 . C = 1 DC = 0

DECF	Decrement	f			
Syntax:	[label] DECF f,d,a				
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f)-1\to d$	est			
Status Affected:	C,DC,N,O	/,Z			
Encoding:	0000	01da	££££	ffff	
Description:	Decrement register 'T. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'F (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1	•			
Cycles:	1.				
Q Cycle Activity:					
Q1	Q2	Q3		Q4	
Decode	Read register 'f'	Proc		Write to destination	
L	<u> </u>			_	

Example:	DE	ECF	CNT,	1,	0
Before Instr	uctio	n			
	=	0x01			
Z	_	U			
After Instru	ction				
CNT	=	0x00			
Z	=	1			

DECFSZ	Decrement f	, skip if 0		DCF	
Syntax:	[label] DEC	CFSZ f,d,a		Syn	
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	$d \in [0,1]$			
Operation:	(f) $-1 \rightarrow \text{dest}$, skip if result = 0				
Status Affected:	None			Sta	
Encoding:	3023	lda ffff	ffff	End	
Description:	The contents of register 'f are decremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (defauit). If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1			W	
Cycles:	1(2)			C	
	Note: 3 cyc	des if skip ar 2-word instru	nd followed action		
Q Cycle Activity	:			Q	
Q1	Q2	Q3	Q4		
Decode	Read register 'f'	Process Data	Write to destination		
If skip:				lf	
Q1	Q2	Q3	Q4		
No operation	No operation	No operation	No operation		
If skip and follo				li	
Q1	Q2	Q3	Q4		
No	No	No	No operation		
operation	operation No	operation	No		
No operation	i	operation	operation		
Example:	HERE	DECFSZ GOTO	CNT, 1, 1 LOOP		
	CONTINUI	Ξ			
Before In:		ss (HERE)			
PC After Instr CNT If CN If CN	ruction = CNT - T = 0; PC = Addre T ≠ 0;				

DCFSN	Z	Decrement 1	, skip if not	0	
Syntax:		[label] DCF	SNZ f,d,a		
Operani	ds: $0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operati	on:	(f) $-1 \rightarrow des$ skip if result			
Status	Affected:	None			
Encodi	ng:	0100 1	lda ffff	ffff	
Descrip	otion:	mented. If 'd' W. If 'd' is 1 the register 'f' (de If the result is	not 0, the ne	is placed in ced back in at instruc-	
		carded, and instead maki tion. If 'a' is selected, ove 'a' = 1, then	already fetch an NOP is exe ng it a two-cyc 0 Virtual bank erriding the BS the bank will b SR value(defa	cuted de instruc- will be SR value. If de selected	
Words	:	1			
Cycles	S :	1(2) Note: 3 cyc by a	cles if skip ar 2-word instru	nd followed	
Q Cyc	ele Activity:	•			
	Q1	Q2	Q3	Q4	
	Decode	Read register "	Process Data	Write to destination	
lf skip);				
•	Q1	Q2	Q3	Q4	
ſ	No	No	No	No	
Į	operation	operation	operation	operation	
lf ski		ed by 2-wor			
	Q1	Q2	Q3	Q4	
	No	No	No operation	No operation	
	operation	operation	No	No ·	
	No operation	operation	operation	operation	
Exa	mple:	HERE ZERO NZERO	DCFSNZ TE	MP, 1, 0	
	Before Inst	3	. ?		
	After Instru	ection			

Unconditional Branch **GOTO** [label] GOTO k Syntax: 0 ≤ k ≤ 1048575 Operands: k → PC<20:1> Operation: None Status Affected: Encoding: $kkkk_0$ k7kkk 1st word (k<7:0>) 1110 1111 kkkk k₁₉kkk kkkk₈ 1111 2nd word(k<19:8>)

Description:

Cycles:

GOTO allows an unconditional branch anywhere within entire 2M byte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-

cycle instruction.

Words:

2

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal "k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example:

GOTO THERE

After Instruction

Address (THERE) PC =

Halt Processor HALT [label] HALT Syntax: None Operands: Processor halts execution after Operation: HALT instruction None Status Affected: 0000 0001 0000 0000 Encoding: While functioning in emulation mode, Description:

execution of the halt instruction will halt processor execution. Toggling the HALT pin or resetting (MCLR = 0) will bring the device out of halt. HALT instruction is not recognized in non-

emulation modes.

Words:

1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	No	No	HALT
	operation	operation	

INCF	Increment f				
Syntax:	[label] INCF f,d,a				
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) + 1 \rightarrow dest$				
Status Affected:	C,DC,N,OV,Z				
Encoding:	0010 10da ffff ffff				
Description:	The contents of register 'f are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				

QC	cle Activity:			
	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process Data	Write to destination

Example: INCF		CF
Before Ins	truction	1
CNT	=	0xFF
Ž	=	0
Ž	=	0 ? ?
DC	=	?
After Instr	uction	
CNT	=	0x00
	=	1
Z C	=	1
DC	=	1

INCFSZ	Increment f, skip if 0		
Syntax:	[label] INCFSZ f,d,a		
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$		
Operation:	(f) + 1 \rightarrow dest, skip if result = 0		
Status Affected:	None		
Encoding:	0011 11da ffff ffff		
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f'. (default) If the result is 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1(2) Note: 3 cycles if skip and followed by a 2-word instruction		

Q Cycle Activity:

Q1	Q2	Q3	Q4
Deco	ie Read	Process	Write to
	register 'f'	Data	destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
Sportant.			

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

operation	operation	operation No	opera
No operation	No operation	operation	opera

Example:	HERE NZERO	INCFSZ:	CNT,	1,	0
	NZERO	:			
	ZERO	:			

Before Instru PC	ction =	Address (HERE)					
After Instruction							
CNT	=	CNT + 1					
	=	0;					
PC	=	Address (ZERO)					
If CNT	#	0;					
PC	=	Address (NZERO)					

Increment f, skip if not 0 INFSNZ INFSNZ f,d,a [label] Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ (f) + 1 \rightarrow dest, Operation: skip if result ≠ 0

None Status Affected:

10da ffff 0100 Encoding:

The contents of register 'f' are incre-Description: mented. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default).

If the result is not 0, the next instruction, which is already fetched, is discarded, and an NOP is executed instead making it a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value

ffff

(defauit).

Words:

1

Cycles:

1(2)

Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	_Q4
Decode	Read	Process	Write to destination
	register 4	Data	desunation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

•	Q1	Q2	Q3	Q4
	No operation	No operation	No operation	No operation
	No operation	No operation	No operation	No operation

Example:

HERE ZERO

REG, 1. 0 INFSNZ

NZERO

Before Instruction

Address (HERE) PC

After Instruction

REG + 1 REG

If REG 0;

Address (NZERO)

Address (ZERO)

Inclusive OR literal with W IORLW

[label] IORLW k. Syntax:

 $0 \le k \le 255$ Operands:

Operation: (W) .OR. $k \rightarrow W$

N,Z Status Affected:

1001 kkkk kkkk 0000 **Encoding:** The contents of W are OR'ed with the Description:

eight bit literal 'k'.. The result is placed in

1 Words: 1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	· ·

Example:

IORLW

0x35

Before Instruction

W 0x9A

After Instruction

0xBF

Inclusive OR W with f IORWF IORWF f,d,a [label] Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ (W) .OR. (f) \rightarrow dest Operation: N,Z Status Affected: ffff ffff 0001 00da Encoding: Inclusive OR W with register 'f'. If 'd' is 0 Description: the result is placed in W. If 'd' is 1 the result is placed back in register T (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

Q2	Q3	Q4
Read register "f"	Process Data	Write to destination
	QZ_	Read Process

Before Instruction

Example:

RESULT, 0, 1

RESULT = 0x13

IORWF

0x91 W

After Instruction

0x13 RESULT = 0x93

Move f MOVF

[label] MOVF f,d,a Syntax:

Operands: $0 \le f \le 255$

 $d \in [0,1]$ $a \in [0,1]$

 $f \rightarrow dest$ Operation:

N,Z Status Affected:

0101 00da ffff **Encoding:**

ffff The contents of register 'f' is moved to a Description:

destination dependent upon the status of 'd'. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value

(default).

1

Words:

1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write W
	register 'f	Data	

REG, 0, 0 MOVF Example:

Before Instruction

0x22 REG 0xFF W

After Instruction

0x22 REG 0x22 W

MOVFF

Move f to f

Syntax:

[label] MOVFF fs,fd

Operands:

 $0 \le f_s \le 4095$

 $0 \le f_d \le 4095$

Operation:

 $(f_s) \rightarrow f_d$ None

Status Affected:

Encodi 1st wo 2nd wo

ing:
rd (source)
ord (destin.)
•

1100 1111	ffff ffff	ffff ffff	ffff _s		
the second secon					

Description:

The contents of source register 'f3' are moved to destination register 'fd'. Location of source 'fs' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination 'fd' can also be anywhere from 000h to

Either source or destination can be W (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVEF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words:

2

Cycles:

2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example:

MOVFF

REG1, REG2

Before Instruction

REG1 REG2

0x33

0x11

After Instruction

REG1 REG2

MOVLB

Move literal to low nibble in BSR

Syntax:

[label]: MOVLB k.

Operands:

 $0 \le k \le 255$

Operation:

k → BSR

Status Affected:

None

Encoding:

0000 0001

Description:

The 8-bit literal 'k' is loaded into the

Bank Select Register (BSR).

Words:

1 1

Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write literal 'k' to BSR

Example:

MOVLB

Before Instruction

0x02 BSR register =

After Instruction

BSR register 0x05

Move literal to FSR LFSR [label] LFSR f,k Syntax: 0 ≤ f ≤ 2 Operands: 0 ≤ k ≤ 4095 $k \rightarrow FSRf$ Operation: Status Affected: None k₁₁kkk OOEE 1110 1110 Encoding: kkkk 0000 k7kkk 1111 The 12-bit literal 'k' is loaded into the Description: file select register pointed to by 'f' 2 Words: 2 Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal	Process Data	Write literal

Example:

LFSR 2, 0x3AB

After Instruction

FSR2H = 0x03 FSR2L = 0xAB MOVLW Move literal to W [label] MOVLW k Syntax: 0 ≤ k ≤ 255 Operands: Operation: $k \rightarrow W$ None Status Affected: kkkk 1110 kkkk 0000 Encoding: The eight bit literal 'k' is loaded into W. Description: Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'K'	Data	

Example:

MOVLW

0x5A

After Instruction

W = 0x5A

MOVWF Move W to f [label] MOVWF Syntax: $0 \le f \le 255$ Operands: $a \in [0,1]$ Operation: $(W) \rightarrow f$ Status Affected: None ffff ffff 111a 0110 Encoding: Move data from W to register 'f. Loca-Description: tion 'f' can be anywhere in the 256 byte bank. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). Words:

Q Cycle Activity:

Cycles:

Q1 (12	Q3	Q4
Decode	Read	Process	Write
	register "f"	Data	register 'f'

Example:

REG, 0 MOVWF

Before Instruction

W 0x4F 0xFF REG

1

After Instruction

W 0x4F 0x4F REG

Multiply Literal with W MULLW Syntax: [label] MULLW $0 \le k \le 255$ Operands: Operation: (W) $\times k \rightarrow PRODH:PRODL$ None Status Affected: kkkk 0000 1101 kkkk Encoding: An unsigned multiplication is carried Description: out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. 1 Words:

1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL

0xC4Example: Before Instruction 0xE2 W PRODH ? PRODL After Instruction

> 0xE2 0xAD PRODH 80x0 PRODL

Multiply W with f MULWF [label] MULWF f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ (W) \times (f) \rightarrow PRODH:PRODL Operation: None Status Affected: ffff ffff 001a 0000 Encoding: An unsigned multiplication is carried Description: out between the contents of W and the register file location "f. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles: Q Cycle Activity:

O1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL

REG, 1 Example: MULWF Before Instruction 0xC4 W REG PRODH 0xB5 PRODL After Instruction 0xC4 0xB5 REG PRODH 0x8A 0x94 PRODL

NEGF		Negate f			
Syntax	C	[label] N	EGF	f,a	
Opera	nds:	$0 \le f \le 25$ $a \in [0,1]$	5		
Opera	tion:	(f)+1	•f		•
Status	s Affected:	N,OV, C,	DC, Z		
Enco	ding:	0110	110a	ffff	ffff
Desc	ription:	plement. T memory lo bank will b	he result cation 'f. e selecte e. If 'a' =	is placed If 'a' is (ed, overric 1, then th	ding the he bank will
Word	is:	1			•
Cycl	es:	. 1			
QC	cle Activity:				
	Q1	Q2	Q3		Q4
	Decode	Read register 'f'	Prod		Write register 'f'
		1			

REG, 1

0011 1010 [0x3A]

1100 0110 [0xC6]

NEGF

Before Instruction

REG

REG

After Instruction

Example:

No Operation NOP [label]. NOP Syntax: None Operands: No operation Operation: Status Affected: None 0000 0000 0000 0000 Encoding: XXXX XXXXX 1111 300000 No operation. Description: 1 Words: 1 Cycles: Q Cycle Activity: Q3 **Q4 Q2** Q1 No No No Decode operation operation operation

Example:

None.

Pop Top of Return Stack POP [label] POP Syntax: None Operands: (TOS) → bit bucket Operation: Status Affected: None 0000 0000 0000 **Encoding:** The TOS value is pulled off the return Description: stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack. 1 Words: Cycles: Q Cycle Activity: Q4 Q3 **Q2** Q1

Decode

POP TOS No No value operation operation

Example:

POP

GOTO

NEW

Before Instruction

TOS

0031A2h 014332h

Stack (1 level down)

After Instruction

TOS PC

014332h

NEW

Push Top of Return Stack **PUSH** [label] PUSH Syntax: None Operands: (PC+2) → TOS Operation: None Status Affected: 0101 0000 0000 0000 Encoding: The PC+2 is pushed onto the top of the Description: return stack. The previous TOS value is pushed down on the stack. This instruction allows to implement a software stack by modifying TOS, and then push it onto the return stack. 1 Words: Cycles:

Q1	Q2	Q3	Q4
Decode	PUSH PC+2 onto return stack	No operation	No operation

Example:

Q Cycle Activity:

PUSH

Before Instruction

TOS = 00345Ah PC = 000124h

After Instruction

PC = 000126h TOS = 000126h Stack (1 level down) = 00345Ah

RCALL	Branch S	ubroutir	10	
Syntax:	[label] P	ICALL	n	
Operands:	-1024 ≤ n	≤"1023"	•	•
Operation:	(PC) + 2 · (PC) + 2 ·	•	C	
Status Affected:	None			
Encoding:	1101	1nm	nnnn	nnnn
Description:	Subroutine call with a jump upto from the current location. First, a address (PC+2) is pushed onto t stack. Then, add the 2's comple number '2n' to the PC. Since the will have incremented to fetch the instruction, the new address will PC+2+2n. This instruction is a temperature.		st, return to the iplement the PC in the next will be	

cycle instruction.

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process Data	Write to PC
	Push PC to stack		
No operation	No operation	No operation	No operation

Example:

HERE

RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address(Jump) TOS = Address(HERE+2)

RESET Reset [label] RESET Syntax: None Operands: Reset all registers and flags that Operation: are affected by a MCLR reset. All Status Affected: 0000 1111 1111 0000 **Encoding:** This instruction provides a way to exe-Description: cute a MCLR reset in software. 1 Words: Cycles:

Q Cycle Activity:

 Q1
 Q2
 Q3
 Q4

 Decode
 Start reset
 No peration
 No operation

Example:

RESET

After Instruction

Registers = Reset Value Flags* = Reset Value

Return from Interrupt RETFIE [label] RETFIE s Syntax: $s \in [0,1]$ Operands: (TOS) \rightarrow PC, Operation: 1 → GIE/GIEH or PEIE/GIEL, if s = 1 $(WS) \rightarrow W$ $(STATUSS) \rightarrow STATUS,$ $(BSRS) \rightarrow BSR$ PCLATU, PCLATH are unchanged. GIE/GIEH, PEIE/GIEL, STATUS reg. Status Affected:

Encoding: 0000 0000 0001 000s

Description: Return from Interrupt. Stack is popped

and Top of Stack (TOS) is loaded into the PC. Interrupts are enabled by setting the either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words:

2

1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example:

RETFIE Fast

After Interrupt

PC = TOS W = WS BSR = BSRS STATUS = STATUSS GIE/GIEH, PEIE/GIEL = 1

Rotate Left f through Carry RLCF [label] RLCF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f < n >) \rightarrow dest < n+1 >$, Operation: $(f<7>) \rightarrow C$ (C) → dest<0> C,N,Z Status Affected: 01da ffff ffff 0011 Encoding: The contents of register 'f' are rotated Description: one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 't' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default). register f C Words: 1 Cycles: Q Cy

ycle Activity:	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

RLCF

REG, 0, 0

Before Instruction

REG C 1110 0110

After Instruction

1110 0110 REG

1100 1100 W

C

Rotate Left f (no carry) RLNCF [label] RLNCF f,d,a Syntax:

 $0 \le f \le 255$ Operands:

 $d \in [0,1]$ $a \in [0,1]$

 $(f < n >) \rightarrow dest < n+1 >,$ Operation:

(f<7>) → dest<0>

Status Affected:

Encoding:

Description:

N,Z

ffff ffff 0100 01da The contents of register 'P are rotated

one bit to the left. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value

(default).



Words:

1

Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

RLNCF

REG, 1, 0

Before Instruction

1010 1011 REG

After Instruction

0101 0111 REG

Rotate Right f through Carry RRCF [label] RRCF f,d,a Syntax: $0 \le f \le 255$ Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<n>) \rightarrow dest<n-1>,$ Operation: $(f<0>)\rightarrow C,$ (C) → dest<7> C,N,Z Status Affected: ffff ffff 00da 0011 **Encoding:** The contents of register 'f are rotated Description: one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). register f Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:

RRCF

REG, 0, 0

Before Instruction

REG = 1110 0110

C =

After Instruction

REG = 1110 0110

W = 0111 0011

C = 0

RRNCF	Rotate Right f (no carry)	
		ı

Syntax: [label] RRNCF f,d,a.

Operands: 0 ≤ f ≤ 255

d ∈ [0,1]

 $a \in [0,1]$

Operation: $(f < n >) \rightarrow dest < n-1 >$,

(f<0>) → dest<7>

Status Affected:

N,Z

Encoding:

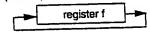
Description:

14,2

0100 00da ffff ffff
The contents of register T are rotated

one bit to the right. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the

bank will be selected as per the BSR value (default).



Words:

1

Cycles:

1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:

RRNCF REG, 1, 0

Before Instruction

REG = 1101 0111

After Instruction

REG = 1110 1011

Example 2:

RRNCF REG. 0, 0

Before Instruction

W =

REG = 1101 0111

After Instruction

W = 1110 1011 REG = 1101 0111

Set f SETF [label] SETF f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ Operation: $FFh \rightarrow f$ None Status Affected: 100a ffff ffff 0110 Encoding: The contents of the specified register Description: are set to FFh. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity: Q4 Q3 **Q2** Q1 Write **Process** Decode Read register 'f' Data register T REG,1 SETF Example:

0x5A

0xFF

Before Instruction

After Instruction

REG

REG

SLEEP		Enter SLEEP mode			
Syntax:		[label] St	EEP		
Operands	s:	None			
Operation	n:	00h →WE	•	_1	
		0 → WDT 1 → TO.	postsc	aier,	
		$0 \rightarrow PD$			
Status A	ffected:	TO, PD			
Encoding	g:	0000	0000	0000	0011
Descripti	ion:	The power-down status bit (PD) is cleared. The time-out status bit (TO) is set. Watchdog Timer and its postscaler are cleared.			
		The proces mode with	ssor is p the osc	ut into S illator str	opped.
Words:	-	· 1	•		
Cycles:		1			
Q Cycle	Activity:				
Q	1	Q2	Q3	(Q4
	Decode	No operation	Proc Da	1	Go to sleep

Example:

SLEEP

Before Instruction

TO = ?

After Instruction

TO = 11 PD = 0

† If WDT causes wake-up, this bit is cleared

Subtract f from W with borrow SUBFWB [label] SUBFWB f,d,a. Syntax: 0≤f≤255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(W)-(f)-(\overline{C})\to dest$ Operation: N,OV, C, DC, Z Status Affected: ffff 010i ffff 01da Encoding: Subtract register 'f' and carry flag (bor-Description: row) from W (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored in register 'f' (default) . If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: 1 Cycles:

Q4

Write to

destination

Q3

Process

Data

Q Cycle Activity:

Decode

Q1

Q2

Read

register 4

Example 1:	SUBFWB REG, 1, 0
Before Instru REG W C	ction = 3 = 2 = 1
After Instruct REG W C Z N	tion = FF = 2 = 0 = 0 = 1; result is negative
Example 2:	SUBFWB REG, 0, 0
Before Instru REG W C After Instru REG W C Z N	= 2 = 5 = 1 ction = 2 = 3 = 1 = 0 = 0 ; result is positive
Example 3:	SUBFWB REG, 1, 0
Before Insi REG W C	= 1 = 2 = 0
After Instr REG W	= 0
C Z	= 2 = 1 = 1 ; result is zero

SUBFWB

Subtract W from literal **SUBLW** [label] SUBLW k. Syntax: Operands: $0 \le k \le 255$ $k-(W) \rightarrow W$ Operation: N,OV, C, DC, Z Status Affected: kkkk kkkk 0000 1000 Encoding: W is subtracted from the eight bit lit-Description: eral 'k'. The result is placed in W. Words: Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W
<u> </u>	literal K	Dala	

0x02

Before Instruction

Example 1:

W С

After Instruction

W

; result is positive 0

SUBLW

CNN

0x02SUBLW Example 2:

Before Instruction

2 W C

After Instruction

W

; result is zero

CZN 0

Example 3:

SUBLW

Before Instruction

W C

After Instruction

; (2's complement) W ; result is negative

CNN

SUBWF Subtract W from f

[label] SUBWF f,d,a: Syntax:

0≤f≤255 Operands:

 $d \in [0,1]$ $a \in [0,1]$

 $(f) - (W) \rightarrow dest$ Operation:

N,OV, C, DC, Z Status Affected:

1

ffff ffff 0101 Encoding: Subtract W from register 'f' (2's com-Description:

plement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register "f" (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per

the BSR value (default).

Words:

1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register "	Process Data	Write to destination

Example 1:	SUBWF REG, 1, 0
Before Instn REG W C	uction = 3 = 2 = ?
After instruc REG W C Z N	
Example 2:	SUBWF REG, 0, 0
Before Inst REG W C	= 2 = 2 = ?
After Instru REG W C Z N	_
Example 3:	SUBWF REG, 1, 0
Before Inst REG W C After Instn REG W C Z N	= 1 = 2 = ? uction
SUBWFB	Subtract W from f with Borrow
Syntax:	[label] SUBWFB f,d,a
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	$(f) - (W) - (\overline{C}) \rightarrow dest$
Status Affecte	
Encoding:	0101 10da ffff ffff
Description:	Subtract W and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

SUBWFB			
Example 1:	st	BWFB	REG, 1, 0
Before Instru	uction		
REG	=	0x19	(0001 1001)
W	=	COx0	(0000 1101)
С	=	1	
After Instruc			
REG		0x0C	(0000 1011)
W	=	0x0D	(0000 1101)
C Z	=	1	, .
N N	=	0	; result is positive
Example2:	s		REG, 0, 0
	a ratio	_	· ·
Before Instr REG			(0001 1011)
W		0x1A	
C	=	0	(0001 1010)
*		J	
After Instru REG		0x1B	(0001 1011)
		0x00	(0001 1011)
W	=		
C Z N	=	i	; result is zero
N	=	0	_
Example3:	:	SUBWFB	REG, 1, 0
Before Inst	tructio	ก	
REG	=	0x03	
W	=	0x0E	(0000 1101)
, C	=	1	
After Instr	uction	l	
REG		0xF5	
W	=	0x0E	(0000 1101)
ç	=		
C Z N	=	0	: result is negative
14	_	•	

Swap f **SWAPF** [label] SWAPF f,d,a Syntax: 0 ≤ f ≤ 255 Operands: $d \in [0,1]$ $a \in [0,1]$ $(f<3:0>) \rightarrow dest<7:4>$ Operation: $(f<7:4>) \rightarrow dest<3:0>$ None Status Affected: ffff ffff 10da 0011 Encoding: The upper and lower nibbles of register Description: 'f are exchanged. If 'd' is 0 the result is placed in W. If 'd' is 1 the result is placed in register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: Cycles: Q Cycle Activity:

Decode Read register 1	Process Data	Write to destination

Example:

SWAPF REG, 1, 0

Before Instruction

REG = 0x53

After Instruction

REG = 0x35

Table Read **TBLRD** [label] TBLRD (*; *+; *-; +*) Syntax: None Operands: if TBLRD *, Operation: (Prog Mem (TBLPTR)) → TABLAT; TBLPTR - No Change; if TBLRD *+, (Prog Mem (TBLPTR)) \rightarrow TABLAT; (TBLPTR) +1 → TBLPTR; if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) -1 → TBLPTR; if TBLRD +*, (TBLPTR) +1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT; Status Affected: None

Encoding:

0000	0000	0000	10nn
			nn=0 *
Ì			=1 *+
1	1	l .	=2 *-
			=3 +*
1 _	l	<u> </u>	

Description:

There are four options with a TBLRD instruction to determine what happens to the 21-bit Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately, and the contents of the program memory location pointed to by the TBLPTR are loaded into the 8-bit Table Latch (TABLAT). The LSb of the TBLPTR selects which byte of the program memory location will be read. If LSb = 1, the high byte will be loaded into the TABLAT. If LSb = 0, the low byte will be loaded into the TABLAT.

TABLA

Words:

1

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (OE goes low) TABLAT updated

Table Read TBLRD TBLRD *+ ; Example1: Before Instruction 0x55 0x00A356 TABLAT TBLPTR MEMORY(0x00A356) 0x34 After Instruction 0x34 TABLAT TBLPTR 0x00A357 TBLRD Example2: **Before Instruction OXAA** TABLAT
TBLPTR
MEMORY(0x01A357)
MEMORY(0x01A358) 0x01A357 0x12 0x34 After Instruction

TABLAT

TBLPTR

0x34

0x01A358

Table Write TBLWT TBLWT (*; *+; *-; +*) [label] Syntax: None Operands: if TBLWT *. Operation: (TABLAT) → Prog Mem(TBLPTR); TBLPTR - No Change; if TBLWT *+, (TABLAT) → Prog Mem(TBLPTR); (TBLPTR) +1 → TBLPTR; if TBLWT *-, (TABLAT) → Prog Mem(TBLPTR); (TBLPTR) -1 → TBLPTR; if TBLWT +*, (TBLPTR) +1 → TBLPTR; (TABLAT) → Prog Mem(TBLPTR); None Status Affected:

0000

Description:

Encoding:

There are four options with a TBLWT instruction. These options determine what happens to the Table Pointer (TBLPTR): no change, post-increment, post-decrement and pre-increment. The current option is determined and the TBLPTR is modified appropriately.

0000

0000

11nn

=1 *+

=2 *-

=3 +*

nn=0 *

The contents of Table Latch (TABLAT) are written to the program memory location pointed to by TBLPTR.

If TBLPTR points to an external program memory location, then the instruction executes in two cycles.

Since the TABLAT is only one byte wide, a multiple of two TBLWT instructions must be executed to program internal memory locations. For example, if the device is defined to program one word at time, an internal memory location is programed in the following manner:

- 1) Set TBLPTR to an even byte
- 2) Write low byte to TABLAT
- 3) Execute TBLWT *+ (2-cycle)
- 4) Write high byte to TABLAT
- 5) Execute TBLWT *+ (long write)

A long write to an internal EPROM location is terminated when an interrupt is received. The post increment TBLWT instruction is the only TBLWT instruction that is recommended for writes to internal memory. (Writes to internal EPROM are only available on devices with 64 or more pins.)

TBLWT	Table Write	
Cycles:	2 (many if long write is to on-chip EPROM program memory)	

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Table Pointer on Address bus)	No operation	No operation (Table Latch on Address bus, WR goes low)

Example1: TBLWT *+

Before Instruction

TABLAT = 0x55 TBLPTR = 0x00A356 MEMORY(0x00A356) = 0xFF

After Instructions (table write completion)

TABLAT = 0x55 TBLPTR = 0x00A357 MEMORY(0x00A356) = 0x55

Example 2: TBLWT

Before Instruction

TABLAT = 0x34 TBLPTR = 0x01389A MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0xFF

After Instruction (table write completion)

TABLAT = 0x34 TBLPTR = 0x01389B MEMORY(0x01389A) = 0xFF MEMORY(0x01389B) = 0x34

Debugger Subroutine Cail RAP [label] TRAP syntax: None)perands: $(PC) + 2 \rightarrow TOS,$ Operation: 000028h → PC<20:1>

INBUG Status Affected:

Encoding:

0000 1110 0000 0000

Description:

Debugger Trap to 00028h. First, return address (PC+2) is pushed onto the return stack. Then the 20-bit value '000028h' is loaded into PC<20:1>. The INBUG status bit is set. TRAP is a two-cycle instruction.

Words:

2

Cycles: Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Push PC to stack	No operation	Write 000028h to PC
No operation	No operation	No operation	No operation

Example:

HERE

TRAP

Before Instruction

Address (HERE) PC =

After Instruction

PC = 0 TOS = A INBUG = 1 000028h

Address (HERE + 2)

TRET	Trap Ret	urn from	Subrout	ine
Syntax:	[label]	TRET		
Operands:	None			
Operation:	(TOS) → PCLATU	PC , PCLATI	H are unc	hanged
Status Affected:	INBUG			
Encoding:	0000	0000	1110	0001
•		-		a stock ic

Description:

Return from debugger trap. The stack is popped and the top of the stack (TOS) is loaded into the program counter. The

INBUG status bit is cleared.

Words:

Cycles:

2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	pop PC from stack
No operation	No operation	No operation	No operation

Example:

TRET

After Interrupt PC = TOS INBUG = 0

Test f, skip if 0 **TSTFSZ** [label] TSTFSZ f,a Syntax: 0 ≤ f ≤ 255 Operands: $a \in [0,1]$ skip if f = 0Operation: None Status Affected: ffff ffff 0110 011a Encoding: If T = 0, the next instruction, fetched Description: during the current instruction execution, is discarded and a NOP is executed making this a two-cycle instruction. If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default). 1 Words: 1(2) Cycles: Note: 3 cycles if skip and followed by a 2-word instruction

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

HERE TSTFSZ CNT, 1

NZERO

ZERO

Before Instruction

PC = Address(HERE)

After Instruction

If CNT = 0x00,

PC = Address (ZERO)

If CNT \neq 0x00.

PC = Address (NZERO)

XORLW	Exclusi	ve OR lite	eral with	W
Syntax:	[label] XORLW k.			
Operands:	0 ≤ k ≤ 2	255		
Operation:	(W) .XOR. $k \rightarrow W$			
Status Affected:	N,Z			
Encoding:	0000	1010	kkkk	kkkk
Description:		tents of W literal 'k'. 7		
Words:	1			
Cycles:	1			
Q Cycle Activity:				•
Q1	Q2 .	Q3-	Q	4

Example:

XORLW 0xAF

Read

literal 'k'

Process

Data

Write to W

Before Instruction

Decode

W = 0xB5

After Instruction

W = 0x1A

(ORWF

Exclusive OR W with f

Syntax:

[label] XORWF f,d,a

Operands:

0 ≤ f ≤ 255

 $d \in [0,1]$

 $a \in [0,1]$

Operation:

(W) .XOR. (f) \rightarrow dest

Status Affected:

N,Z

Encoding:

0001 10da ffff ffff

Description:

Exclusive OR the contents of W with register 'f'. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in the register 'f' (default). If 'a' is 0 Virtual bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value

(default).

Words:

1

Cycles:

1

Q Cycle Activity:

Ω1	Q2	Q3	Q4
Decode	Read	Process	Write to
Decodo	register 'f'	Data	destination

Example:

XORWF REG, 1, 0

Before Instruction

 $\begin{array}{rcl}
\mathsf{REG} & = & \mathsf{0xAF} \\
\mathsf{W} & = & \mathsf{0xB5}
\end{array}$

After Instruction

REG = 0x1A

W = 0xB5